ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/281833874

Turing test and intelligence with trick

Conference Paper · September 1997

CITATIONS
9

READS
464

2 authors:

Zhaohao Sun
Papua New Guinea University of Technology
287 PUBLICATIONS **2,600** CITATIONS

SEE PROFILE

Klaus Weber
Robert Bosch Manufacturing Solutions, Stuttgart
28 PUBLICATIONS **251** CITATIONS

SEE PROFILE

All content following this page was uploaded by Zhaohao Sun on 02 May 2016.

The user has requested enhancement of the downloaded file.

This paper publication information is as follows.

Sun, Zhaohao and Weber K (1997) Turing test and intelligence with trick. In: Proc 8th Ireland Conf on AI (AI-97), Londonderry, Ireland; September. pp 217-224

If you use it for research, please use the above information for citation and referencing

Thank you

Dr Zhaohao Sun

Turing Test and Intelligence with Trick

Sun Zhaohao, Klaus Weber
(Fakultät für Mathematik, Informatik und Naturwissenschaften
TU Cottbus, 03013 Cottbus Germany)

Abstract. In this paper we investigate the Turing test and its related software development. We also analyse the tricks involved in the Turing test and in the programs for passing the Turing test. Based on logic with trick, we discuss the necessary condition of a Turing system to pass the Turing test. Finally we present a design for a Turing systems, based on the architecture of CHATTERBOT [1]. We believe therefore that the serious research into intelligence with trick can help not only improving the programs for passing the Turing test, but also better understanding human real intelligence as well as developing artificial intelligence.

Keywords: Turing test, Artificial intelligence, Intelligence with trick, the Loebner contest, Logic with trick.

1. Introduction

As we know, A. Turing proposed the imitation game, known as the Turing test, in 1950 [1, 2]. He called it as a replacement for the question "Can machines think?" Turing believed that in about 50 years technological progress would produce computing machines with a storage capacity of 10^9 bits, and with such machinery, a computer program would play the Turing test so well that an average questioner would not have more than 70% chance of making the right identification after five minutes of questioning [2]. Hence, if a computer program could pass the Turing test, then it could be said to be exhibiting intelligence. No doubt, the Turing test and literature [2] led to the serious research and development of AI such as natural language translation, game playing, automatic theorem proving, machine learning and expert system in the following almost half a century [10, 13]. However, we have to admit two facts:

1. Although a few attempts had been made in order to understand and pass the Turing test in the past, for example the ELIZA [4], PARRY [5] SHRDU [6] and CHATTERBOT [1], the related programs are no doubt still a long way for passing the Turing test in an unrestricted way.
2. We have not yet studied the trick involved in the Turing test seriously [1, 2, 4, 7]. Even now, some still believe that it was shameful to study the trick or intelligence with trick [4], although it is an important component of human real intelligence [7]. Many have been yet unaware of the intelligence with trick involved in the Turing test, although a few such as Mauldin [1] and Hutchens [4] have already asserted that when someday a computer program does pass the Turing test, it will use many tricks, for the simple reason that people already use them everyday. Besides, intelligence with trick is much more difficult to be studied than the intelligence based logic and intelligence based computation which constitute the history of artificial intelligence. For more detail also see [7] and our forthcoming paper [8].

In this paper we overview the Turing test and its related software development. We also analyse the tricks involved in the Turing test and in the Turing systems[1]. Based on the logic with trick (see section 7), we then investigate the necessary condition for a Turing system to pass the Turing test in the Loebner contest. Finally we present a design for a Turing systems, based on the architecture of CHATTERBOT [1]. Therefore we believe that the serious research into intelligence with trick can support not only the improvement of a program for passing the Turing test, but also a better understanding for human real intelligence as well as the development of artificial intelligence.

2. Turing Test and its Development

The basis of the Turing test is simple. A questioner can communicate with two subjects by typing messages on a computer terminal. The questioner knows that one subject is a human being and that the other is a computer program, and its task is to guess which is which. The computer program tries to fool the questioner into making the wrong identification, while the human being assists the questioner to make the correct identification.

One of the great advantages of the Turing test is that it allows the questioner to test almost all of the evidences that we would assume to constitute thinking [4]. In 1965, Weizenbaum's ELIZA program, a conversation system, demonstrated that a simple computer program could successfully play the Turing test by resorting to a few "tricks," the most important participation to answer questions with questions [1]. ELIZA consists of two stages [4]. The first stage uses a parser to extract information from what was typed to the computer, whereas the second stage uses a script to formulate a suitable reply. ELIZA attracted a great deal of interest at that time. However, the more interesting result was Colby's work on PARRY [5]. Colby's aim was to verify a theory of the paranoid mode of behaviour by using the theory to build a computer simulation, and then using the Turing test to determine whether an expert psychiatrist could distinguish between the program and an actual paranoid patient [4]. PARRY consists of a parsing module and an interpretation-action module. The former attempts to understand the interview input of clinicians communication in unrestricted natural language, and the meaning of the input is passed to the interpretation-action module. The latter is made up of production rules that size up the current state of the interview and decide which linguistic actions to perform to fulfill the models intentions. What differs from ELIZA is that PARRY has knowledge about the conversation so far, and its current state of mind [4, 5]. Colby's Turing test indicated that expert psychiatrists were unable to do better than chance in distinguishing the ramblings of PARRY from transcripts of conversations with real paranoids. In our opinion, PARRY is an advance over ELIZA, because PARRY has a personality [1].

Winograd's SHRDU is probably the most advanced of these early systems [4, 6]. The program simulates a robot which can interact with a simple world which consists of coloured building blocks and boxes on a flat surface. SHRDU has knowledge about this world, and it can answer questions about it posed in natural language. It is also able to perform actions on this world and has the ability to learn about new tasks. Winograd created SHRDU as an example of how a natural language understanding system should

1. For simplicity, we refer a program for passing the Turing test as a Turing system in the whole paper.

be created. He believed that a computer cannot deal reasonably with language unless it can understand the subject it is discussing. In modelling language understanding, all aspects of language must be integrated. Hence SHRDU was given knowledge about its world and itself, and the parser, recognition grammar, semantic analysis and problem solving modules are also integrated in it. Although his SHRDU wouldn't have a chance in winning the Turing test, because of its limited domain of discourse, Winograd has approached the problem in a sensible way [4].

As an elegant reimplementation of Richard Bartle's multiuser dungeon (MUD), TINYMUD was opened in 1989 [1]. Its key features include: multiplayer conversation, scenery simulating physical spaces such as towns and buildings, and user extensibility. The last feature, the ability of players to create their own subareas within the world model, was a key feature that made TINYMUD very popular. TINYMUD provided a world filled with people who communicate by typing. This seemed to us to be a ripe opportunity for work on the Turing test, because it provided a large pool of potential judges and interviewers. In TINYMUD, computer controlled players are called bots, short for robots. Many simple robots were created, and even ELIZA was connected to one stationary robot. Using TINYMUD, Mauldin created a computer controlled player [1], a CHATTERBOT, that can converse with other players, explore the world, discover new paths through the various rooms, answer players' questions about navigation and answer questions about other players, rooms and objects. They included a lot of humour among the responses, and succeeded in making an agent more interesting than ELIZA.

Although a few attempts had been made in order to understand and pass the Turing test before 1991, the related programs are no doubt still a long way for passing the Turing test in an unrestricted way. Because one has not been yet aware that the intelligence with trick involved in the Turing test is much more difficult to be studied than the intelligence based on logic and intelligence based on computation which constitute the history of artificial intelligence.

3. The Loebner Prize and the Loebner Contest

The Cambridge Centre for Behavioural Studies started the Loebner Prize in 1991: an annual contest between computer programs to identify the most "human" programs, and eventually to award US $ 100,000 to the program that first passes an unrestricted Turing test [1,3,4].

The first Loebner contest was held on November 8, 1991 in Boston's Computer Museum. In this contest, six computer programs were accepted as subjects. Four human subjects and ten judges were selected from respondents to a newspaper advertisement, and had no special expertise in computer science. However, this contest has a few shortcomings and therefore was really only a restricted version of the Turing test, because the Loebner contest prize committee was aware that the computer programs didn't stand a chance, and then restricted the topic of conversation on each terminal [4]. In this contest, interestingly, several computer programs were mistaken for human beings by a few of the judges. At the same time, one of the human beings was mistaken for a computer program. The contest was smaller in the following years. In 1992 and in 1993, down to three computer programs from six, and the same three programs have finished first, second, and third all three years. In the 1994 contest the domain of dis-

course was also restricted. The TIP was the winner because of suiting perfectly to this restriction in that year. The 1995 Loebner contest was unrestricted, that is, the judges could talk to the subjects about anything whatsoever, and were free to ask trick questions. Furthermore, all judges involved in the computer industry and possessed the skills needed to fool the computer programs into revealing their identity. As a result, the computer programs in this contest performed considerably worse than in previous years [4].

Of these five contests, Weintraub's PC Therapist has won the contest four times and Whalen's TIP system has won the contest once. PC Therapist is probably nothing more than an ELIZA derivative. TIP provides information on a particular topic in response to queries in natural language. It is therefore more of a database system than a conversation system. After lost in the 1995 Loebner contest, Whalen believed the best approach was to attempt to model a human being, to the extent of including a personality, personal history and world view within the program. Besides, the purpose of Weintraub's program is to simulate conversation whereas Whalen develops information system for a particular topic [4].

In sum, in the past Loebner contests, the contestants limited the conversational possibilities in some way. Whalen restricted the conversation to his little drama. Weintraub simulated whimsical conversation which can be meaningless by definition. PARRY simulated a paranoid by using nonsequiturs and by being unresponsive. These were all creative and devious ways of restricting what the judges could talk about [4]. Of course, the current work on the Loebner contest will hasten the passing of an unrestricted Turing test. Any such program will have to be more than just a general problem solver: some of its knowledge and tactics will be based on long experience with typed communication with real people.

4. Some Considerations to the Turing System

Almost half a century have already elapsed since the Turing test was proposed. Computers have become significantly more powerful, while storage space and processing performance have increased exponentially. The field of AI and expert systems has blossomed, and researchers are working on problems such as image and speech recognition. Given this, we may very well wonder why no computer program has yet passed the Turing test [1,4,8]. At least as far as the entrants of the Loebner contest go, the complexity and capability of such systems have remained almost where they were thirty years ago when ELIZA was created [4].

We firmly believe that what is the most important for a Turing system is it must simulate real conversation as much as possible. At the same time, the contestants must learn that what his Turing system confronts is a real human being or human expert with the intelligence, of which intelligence with trick is an important component. In other words, it is very difficult for a contestant to win the Loebner Contest without thorough research into the human's trick and plot exhibiting in a real conversation. In fact, trick has already been used in many programs to some extent, in order to pass the Turing test. So Shieber has criticized the Loebner contest as a parlour game, rewarding tricks rather than furthering the field of AI [3]. However, if we study the Turing test seriously, we will have to admit the fact that if A. Turing lived today, he would have actively supported to use some kinds of tricks in program in order to pass the Turing test. There-

fore, in what follows, we will discuss the trick in the Turing test and in the related Turing systems.

5. Tricks in the Turing test

As we knows, the reason why Turing introduced the Turing test was only to replace the question "Can machines think?" with the imitation game, because he believed that the meaning and answer to the question "can machines think?" was absurd and too meaningless to deserve discussion [2], although that game could be expressed in relatively unambiguous words [2]. It should be said that from fuzzy logic viewpoint, unambiguous does not mean "precise". It maybe mean only less ambiguous. It is just this "less ambiguous" words that have not made us understood for almost half a century, because we have not yet used programs to pass the Turing test in an unrestricted way successfully. In fact, from [2] we can see that Turing only searched all strategies not only of rational intelligence but also of intelligence with tricks to reason the feasibility of his conjecture or predication about the Turing test. In what follows, we can see how Turing referred to tricks in his paper [2].

1. "It is natural that we should wish to permit every kind of engineering technique to be used in our machines." [2] We hence can use logic and reasoning with trick [7, 8] in a program or an expert system in order to pass the Turing test in an unrestricted way.
2. He proposed that "a computer program would be *fool* the average questioner within fifty years"[1, 2]. What is "fool"? We think, fool means to use trick. Moreover, an average questioner possesses a great deal of intelligence with trick besides rational intelligence.
3. " It is A's (computer) object in the game (the Turing test) to try and cause C (questioner) to make the wrong identification" [2][1]. So the Turing test must use intelligence with trick to answer what the questioner asked. Hence Turing did not opposite to use trick.
4. Turing believed that the machine (programmed for playing the Turing test) would *deliberately* introduce mistakes in a manner calculated to confuse the questioner [2]. In other words, Turing hinted us how to use intelligence with trick to confuse the questioner.

Based on the above analysis, we can assert that Turing also *hinted that it was impossible* to make a Turing system successful without serious and rigorous study on intelligence with trick and ingenious use of intelligence with trick. Therefore, it is strange why we haven't had courage to study intelligence with trick yet, considering that we have studied almost all what Turing commended in [2] over the past 50 years, for example, natural language translation, game playing, automatic theorem proving, machine learning and expert system [2, 10].

By the way, in the Turing test there is also fuzziness. For example, "average" or "average questioner" is a fuzzy concept. From a linguistics viewpoint, the significance of an average (intelligent) questioner is always dependent on syntax, semantics and pragmat-

1. The words in brackets were inserted by authors.

ics. Even that we may think that the significance change of an average questioner is less dependent on syntax and semantics, it is still strongly dependent on pragmatics which is in fact a human-culture-society network [9]. In other words, an average questioner in the 1950's was very different from that in the 1990's, in particular, with respect to the possession of the knowledge of computer science and technology. Therefore it is nowadays more difficult for a computer program to fool an average questioner into making the wrong identification in the Turing test than in the 1950's in some cases. This is also one of the reasons why there is not essential progress in passing the Turing test.

6. Tricks in the Turing Systems

As we mentioned above, tricks have already been used in many programs such as ELIZA and PARRY to some extent, in order to pass the Turing test. Now we turn to analyse these tricks involved in some programs.

ELIZA's tricks [1]: ELIZA's main trick was to use questions to draw a conversation out of the user himself, with little or no actual contribution from the program. This works because most people like to talk about themselves, and are happy to believe the program is listening. A further trick is the use of the Rogerian mode, which provides unimpeachable cover for the computer. Since the program never says anything declaratively, it cannot contradict itself later.

PARRY's tricks [5]: PARRY has a much larger collection of tricks: admitting ignorance, i.e. "I don't know"; changing the level of the conversation, i.e. "Why do you ask that?"; rigidly continuing a previous topic; introducing a new topic and launching into a new story.

CHATTERBOT's tricks [1]: The CHATTERBOT implements all of the above tricks from ELIZA and PARRY. It also uses several other tricks:

- Having many fragments of directed conversation stored in an activation network. These fragments enhance the "connectedness" of the program's own responses, and if the user plays along, the whole conversation can hang together for a while.
- Controversial statements are a way to drag the user into the program's conversation, rather than letting the user direct the discourse. In fact, in a human conversation, usually a man with a special intelligence can control the whole conversation, or he can lead the conversation to the direction benefited for him. Research into this kind of intelligence is also necessary for the contestant.
- Humorous statements also make the program seem more human.
- The program can safely agree with the user, in certain cases, rather than being non-committal.
- Simulating typing. Imitating the rhythm of a person typing and slowing down the interaction also gives the program fewer chances to make a big mistake.

In his design of the Turing system entering the Loebner Contest in 1996, Hutchens confessed that if no database match was found, his program would then try to detect whether the sentence was a trick sentence [4]. It would look for common ways judges used to trick programs (such as asking them mathematical questions or typing non-

sense words), and respond with suitably witty replies (this can be an application in expert system or in database).

In sum, if an average questioner uses many tricks to converse with a Turing system in a Loebner contest, it is thus impossible for the program to defeat this questioner, if the program has no ability to recognize, to analyse the related tricks and to use corresponding tricks to resolve the related tricks. Hence, when someday a computer program does pass the Turing test, it will use many tricks, for the simple reason that people already use them everyday. For example, a child of 3 years can use tricks to fool his parents [7]. Therefore recognition, analysis and utilization of tricks in a program is never a shameful thing [1] but a appropriate action for hastening the further development of AI [8], if one of the aims of AI is still to understand and simulate human real intelligence.

It should also be said that from the above discussion we can see all tricks used in the related Turing test essentially followed Turing's suggestions in [2]. In our opinion, it is necessary to go a step forward after implementing all above-mentioned tricks. More specially, by analysing the typing speed and typed content of the user, we can further analyse his personality and his past experience history to classify him into one of the different classes which correspond to different tricks, because the trick is usually dependent on one's personality and his past experiences. In other words, we must design an expert system with trick (as a subsystem of the related program) to classify the tricks into different classes and investigate the relation of tricks and one's personality and his past experiences. Moreover, a trick base, recognition of trick, trick resolution and an inference mechanism with trick are main components of the expert system with trick. We think this is a necessary condition for a program to pass the Turing test (also see section 8). On the other hand, adding real world knowledge and a deeper understanding on human real conversation in addition to the tricks will no doubt help not only the program to improve in the coming Loebner contest, but also to understand human real intelligence better [1].

7. The Turing Test and Logic with Trick

After serious analysis on the development of programming and storage space as well as human intelligence (or mind), Turing believed that in about fifty years' time it would be possible to make a computer program to pass the Turing test [2]. However, based on his analysis on the software development for passing the Turing test and experiences of participation in the Loebner contest, Hutchens believed that a program for passing the Turing test in the Loebner contest would doomed to failure [4]. In what follows, we discuss under which necessary conditions a Turing system can pass the Turing test in an unrestricted way from logic with trick [11] and expert system viewpoint [13].

As we know, a formal logic consists principally of a language set L and an inference rule set I. In classical logic, for example, propositional logic and predicate logic, usually one uses a fixed language set and a fixed inference rule set to discuss its logical properties or to carry out the related logical reasoning. Similarly, an expert system principally consists of the knowledge base and the inference engine [12]. The knowledge base contains all the knowledge about a certain domain that has been processed by knowledge engineers and/or the domain experts. The inference engine contains many decision and inference strategies to do reasoning using the knowledge from the knowledge base. From a logic viewpoint, the knowledge base and the inference engine can

hence be considered as the generalization of the L and I of a logic < L, I > respectively. In other words, an expert system can be considered as a generalization of a logic system < L, I>, as shown in figure 7.1.

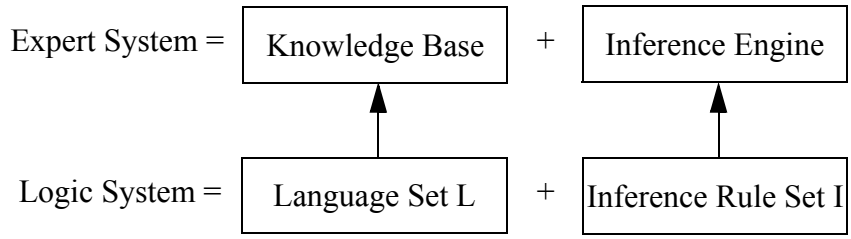

Fig. 7.1 The relation between a logic system and an expert system

With the development of expert systems, one has to discuss the changeable knowledge base or changeable inference engine, because the knowledge base can be exchanged in some expert systems [12]. In particular, after running for some time, the knowledge in the knowledge base of such expert systems will be often updated, expanded and modified. So a knowledge base in such expert systems is dynamic. In order to cope with this dynamic case, we like to introduce the notion of logic with trick in a most general sense and then discuss the problem we introduced earlier.

Definition [11]. Let <L, I > be any fixed logic, L' and I' be the superset or subset of the language set L and the inference rule set respectively, then we call the logic <L', I>, <L, I'>, <L', I'> *type 1 logic with trick, type 2 logic with trick, type 3 logic with trick* with respect to the logic <L, I> respectively. For convenience, usually, one calls any of these logics *logic with trick* in a unified way.

Based on the above analysis, we can now consider a running expert system a generalization of the type 1 logic with trick < L', I > with respect to logic < L, I > which corresponds the original expert system. It should be said here, because the artificial intelligence researchers always see the dynamic situation of an expert system with the development in its knowledge base. Thus they cannot be able to see the tricks exhibiting in the expert system corresponding to type 1 logic with trick <L', I>. By contrast, we stand in the world of the logic < L, I > and can clearly see which tricks has appeared in < L', I >. This is one of the reasons why we consider the < L', I > as a type 1 logic with trick.

Because the knowledge base and the inference mechanism of a Turing system by a Loebner contest are "static", and the knowledge base and the thinking strategies of a questioner in the Loebner contest however are dynamic, we can consider the Turing system in the Loebner contest as a generalization of a base logic < L, I >, the questioner as a generalization of a type 3 logic with trick with respect to < L, I >, as shown in figure 7.2. More generally, we may suppose that L, I are the knowledge set of the knowledge base and the inference strategies set of the Turing system, L' and I' the knowledge set and question strategies of the questioner respectively. Hence we can view Turing system as <L, I>, the questioner as <L', I'> respectively. In what follows, based on the above assumption we turn to investigate the necessary condition for a Turing system to pass the Turing test in an unrestricted way. In general, if we only know $L \not\subset L'$ or $L' \not\subset L$ or $I \not\subset I'$ or $I' \not\subset I$ [1], then we cannot decide the necessary condition for a Turing system to pass the Turing test, because which one can win depending

strongly on the time and the involved topics in the questioning under these cases, either of both has a chance to win in the Turing test. But the following four case are regular. It is necessary to investigate them in detail.

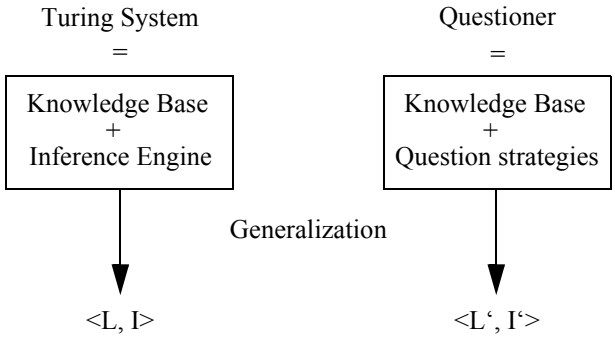

Fig. 7.2 The relation of a Turing system and a questioner by the Loebner Contest

case 1. $L \subseteq L'$, $I \subseteq I'$

case 2. $L \subseteq L'$, $I' \subseteq I$

case 3. $L' \subseteq L$, $I \subseteq I'$

case 4. $L' \subseteq L$, $I' \subseteq I$

From a logic viewpoint, if case 1 is valid, then the questioner has the advantage of possessing both the knowledge and the using the inference strategies of the Turing system. Hence, the Turing system may be defeated. This is an ideal goal that the questioner tries to get. Similarly, if case 4 is valid, then the questioner may be defeated. This is the ideal goal for the designer to let his Turing system get. These two cases are simple. However, it is difficult to let case 4 be valid, because usually the average questioner is very trickish in the contest, thus it is difficult to make either $L' \subseteq L$ or $I' \subseteq I$ valid in a Turing system. Case 1 is the normal case for the moment. This is the reason why all Turing systems have not passed the Turing test yet. Case 2 appears seldom in the Turing test, the reason is similar to case 4. Case 3 is sometimes valid in the Turing test. But this case is some complicated to analyse. Because in a special time and a special domain, it is possible for a Turing system to possess more knowledge than the questioner has, at the same time, the questioner will try to use trick as much as possible in order to defeat the Turing system. So, it is difficult to decide or evaluate who could be the winner. This is the reason why a Turing system can be defeated at a special time without regards of statistics analysis (also see section 3). However, in usual, a Turing system doesn't possess more knowledge than an average questioner. In other words, usually a Turing system is a specialized knowledge-based system. So it is brittle [13]. That is, it cannot cope with novel situations, and its performance degradation is not graceful. Therefore a very large knowledge based system aimed at capturing human commonsense knowledge [13] and resolving the knowledge with trick and reasoning with trick [7] is a necessary condition for a Turing system to pass the Turing test. In other words, L' and I' have to become the subset of L and I in a fixed time respectively. This is the necessary condition for a Turing system to pass the Turing test.

1. Similarly, we only know $L' + L \neq \varnothing$ or $I' + I \neq \varnothing$. Here "+" stands for the disjoint sum of sets.

In sum, from a logic viewpoint, we have used type 3 logic with trick to investigate the necessary condition of a Turing system to pass the Turing test. We believe that this analysis will provide a basic and important idea for the future development of a Turing system.

8. Design of a Turing System

Based on the above discussion, we present now an enhancement of Mauldin's CHATTERBOT [1] by an expert system with trick as shown in figure 8.1. It consists of three subsystems assisting an inference engine with trick. The underlying idea is personality being an ensemble of thoughts, feelings and deeds characterizing a person. Thus the system has to gain as much facets of these spheres as is possible by just analysing the mode and content of its disputant's typing. Of course, in the Turing test, "deeds" are only the way of typing and the expression of thoughts and feelings in what is written. The most difficult thoughts to handle are tricks. So the trick knowledge base (also see figure 8.1) should include as many tricks as possible to enable the recognition and classification of current appearing tricks and to respond in a proper way. Besides using tricks there are still other "dangers" for the Turing system: its conversation partner could be an expert, which might lead to inappropriate answers or comments of the system. So the system has to be capable of deciding whether this is the case. This task requires a world knowledge data base, i.e. a vast amount of subjects, sciences and other "hard knowledge" as well as e.g. data on the last Olympic games and such matter for chatting and gossiping. Due to this data base, the system would be able to estimate the judge's knowledge and to classify him as an layman or an expert or others. If the system couldn't keep up technically with the conversation it may either permit this lack or use strategies to draw the conversation to another topic which might interest the judge without arresting his attention. In the Loebner contest the judge already knows that his disputant may be a computer. Hence he would hardly show feelings which the Turing system should take into account, but try to fool it. However, if he is not too prejudiced the Turing system can get a profit from analysing his feelings. The result of this evaluation shows the system if it is rewarding to continue talking about a sad subject or just euphorically chatting about a matter of "common" interest. Supported on the mentioned information the inference engine with trick (see figure 8.1) produces a reply.

9. Conclusion Remarks

In this paper we have investigated the Turing test and some programs for passing the Turing test and analysed the tricks involved in the Turing test and in the related programs. Interestingly, we had to assert that A. Turing recommended, in fact, to use tricks in designing a program in order to pass the Turing test. Furthermore, we believed that adding real world knowledge and deeper understanding on human real conversation in addition to tricks will no doubt help not only a program to improve in the coming Loebner contest, but also to understand human real intelligence better. In the last we have used type 3 logic with trick to investigate the necessary condition of a Turing system to pass the Turing test and presented a design for a Turing system, based on CHATTERBOT [1]. We believed that a very large knowledge based system armed at capturing human commonsense knowledge [13] and resolving the knowledge with

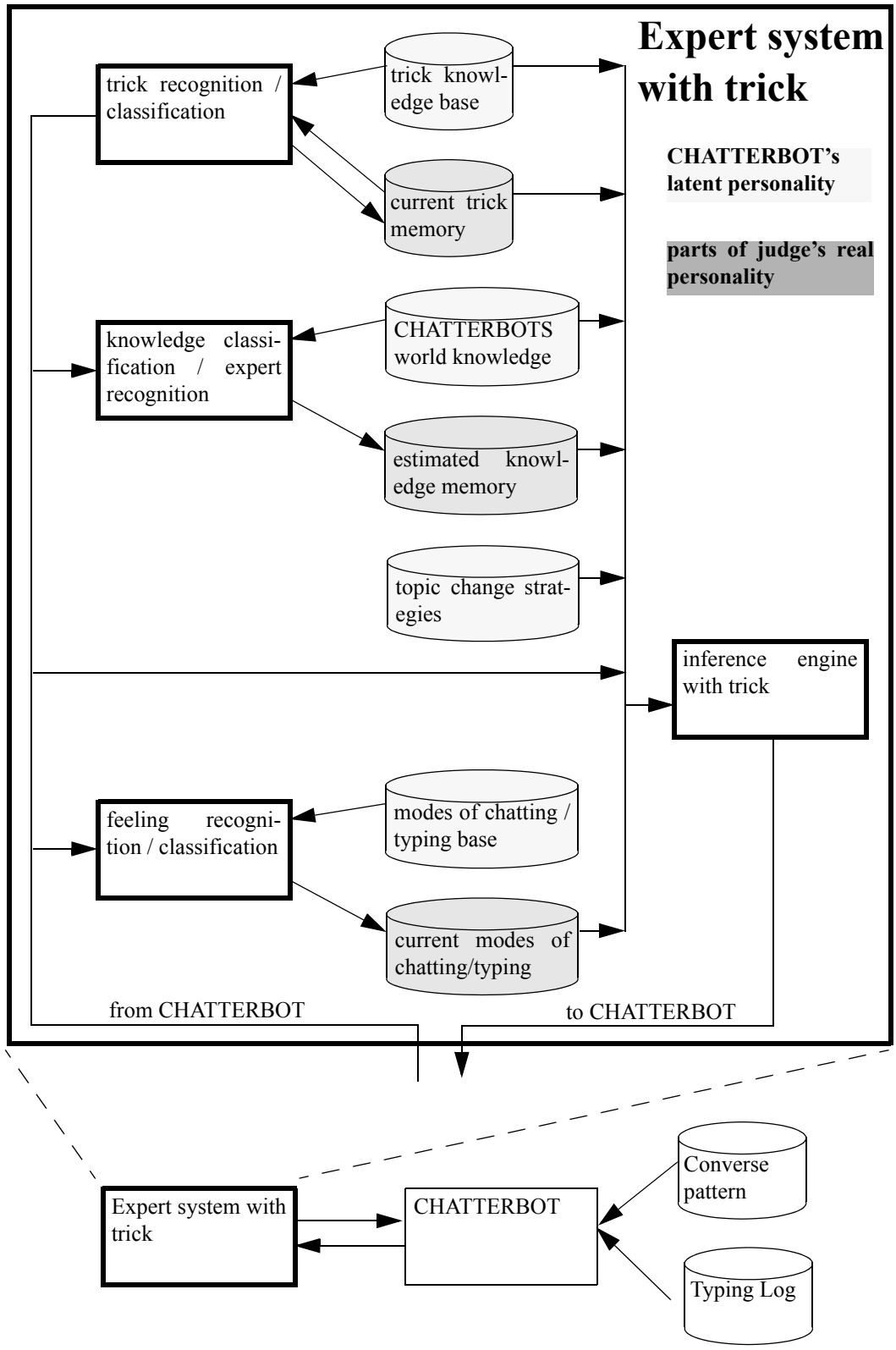

Fig. 8.1 Architecture of a Turing system

trick and reasoning with trick [7] is a necessary condition for passing the Turing test, which is basic and important for the future development of a Turing system.

References

[1] M. L. Mauldin: Chatterbots, Tinymuds, And the Turing Test: Entering the Loebner prize Competition, Proc. AAAI-94, USA, August 1994

[2] A. M. Turing: Computing machinery and intelligence, Mind, Vol.54 No.236 Oct. 1950, pp. 433-460

[3] S. Shieber: Lessons from a restricted Turing test, Tech. Rep. TR-1992, Harvard Univ. Sept, 1992.

[4] J. I. Hutchens: How to pass the Turing test by cheating, TR 97-05 The Centre for Int. Inf. Proc. Sys., The Univ. of Australia, 1997

[5] K. Colby: Artificial Paranoia: *A computer simulation of paranoid process*, Pergamon Press, New York, 1975

[6] T. Winograd: Understanding natural language, Academic press, 1972

[7] Sun, Zhaohao: Intelligence with trick, CADM'94 Beijing, 1994

[8] Sun, Zhaohao; Klaus Weber: Outline of intelligence with trick, working paper, TU Cottbus, 1997

[9] Qian, Guanlian: Pragmatics: human culture society networks, Reading (in Chinese), No.11, ISSN 0257-0270, 1996

[10] Sun, Zhaohao: AI development and exploration to some AI issues, Vol.14. No.1, J. Hebei Univ. 1994

[11] Sun, Zhaohao: Klaus Weber: Logic with Trick, working paper, TU Cottbus, 1997

[12] Zimmermann, H.-J.: Fuzzy set theory and its applications, Kluwer Academic Publishing, 1991

[13] Rich, E. Knight, K: Artificial Intelligence, McGraw-Hill, Inc. 1991

View publication stats